# At The Beginning…

Heaven is compiled line by line. There is no need for any semicolons, which is why they have been omitted.

# Datatypes

| Type | Refers To |
|---|---|
| text | String |
| num | Integer |
| lnum | Long |
| fnum | Float |
| truth | Boolean |
| listastext | ArrayList<String> |
| listasnum | ArrayList<Integer> |
| listaslnum | ArrayList<Long> |
| listasfnum | ArrayList<Float> |
| listastruth | ArrayList<Boolean> |

**null** and **void** keywords are represented with **empty**.

# Variable Assigning

Allowed patterns:

@type var = empty
@type var = val
@type var1, var2, var3 = val1, val2, val3

Multiple assigning at once is allowed.

# Functions

## Main Function

$type name @para
//code
$done

## Void Function

$empty name @para
//code
$done

## Other Functions

$type name @para
//code
$return val

If the value is text and represented with a variable name, the underscore mark ( _ ) must be put before it. Except, there is no need to use double quotes for texts.

# Loop Statements

Instead of **for** and **while** keywords, only **loop** is used in Heaven.

## For Loop

loop type var = val; condition; iteration
//code
end

## While Loop

loop condition
//code
end

## For-Each Loop

loop type var in array
//code
end

# Conditional Statements

## If

if condition
//code
end

## Else If

elif condition
//code
end

## Else

else condition
//code
end

## Switch - Case

**acc** means **according to**.

acc var
let val1
//code
block
let val2
//code
block
let val3
//code
block
finish

# Lists

Allowed patterns:

@type list1 = {item1, item2, item3}
@type list1, list2, list3 = {item1, item2, item3} {item4, item5, item6} {item7, item8, item9}

If the type is **text,** there are some extra rules:

- Text values are written without quote marks.
- To put text values, if they include, comma (,) and closing curly bracket ( } ) must be written with ‘ \ ’ symbol.
- If a variable will be added instead of text value, the underscore mark ( _ ) is put on its name.

# IO Library

IO library is used for basic input - output functionality.

## Output

io >> out “val”
io >> out var

## Input

io >> in = @type var
io >> in = var

# Lists Library

**Lists** library provides basic functions for lists.

## Set Item

lists >> set at index in list _var (for any type)
lists >> set at index in list val (no need for double quote marks)

## Get Item

lists >> get at index in list = @type var
lists >> get at index in list = var

## Add Item

lists >> add in list _var (for any type)
lists >> add in list val (no need for double quote marks)

## Remove Item

lists >> remove at index in list